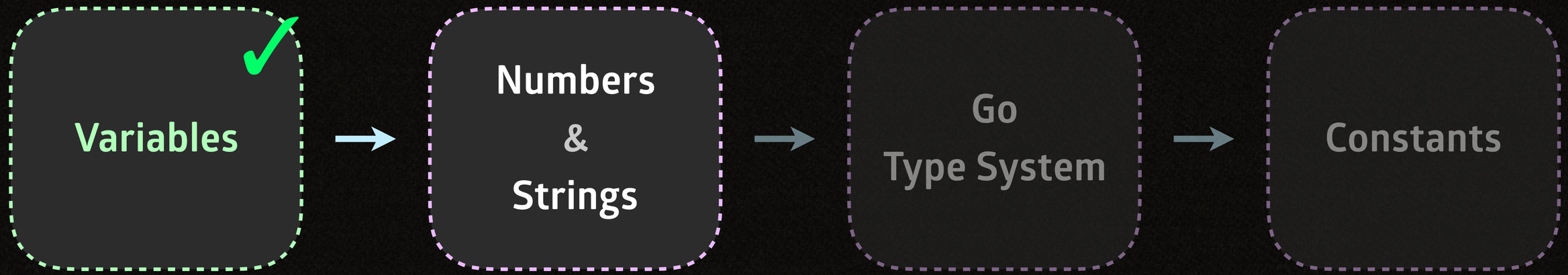
PART II

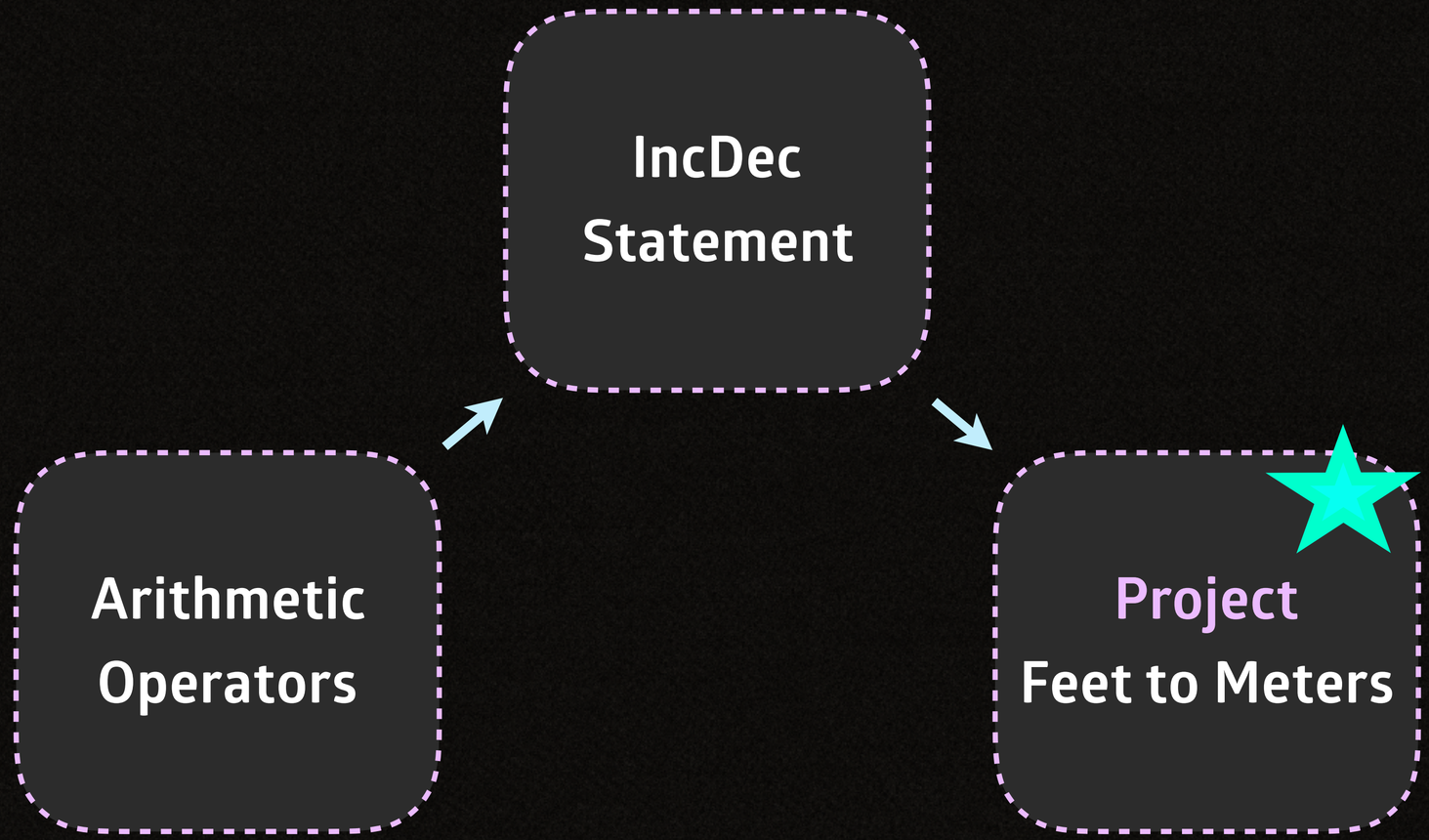
Fundamentals

Representing Data



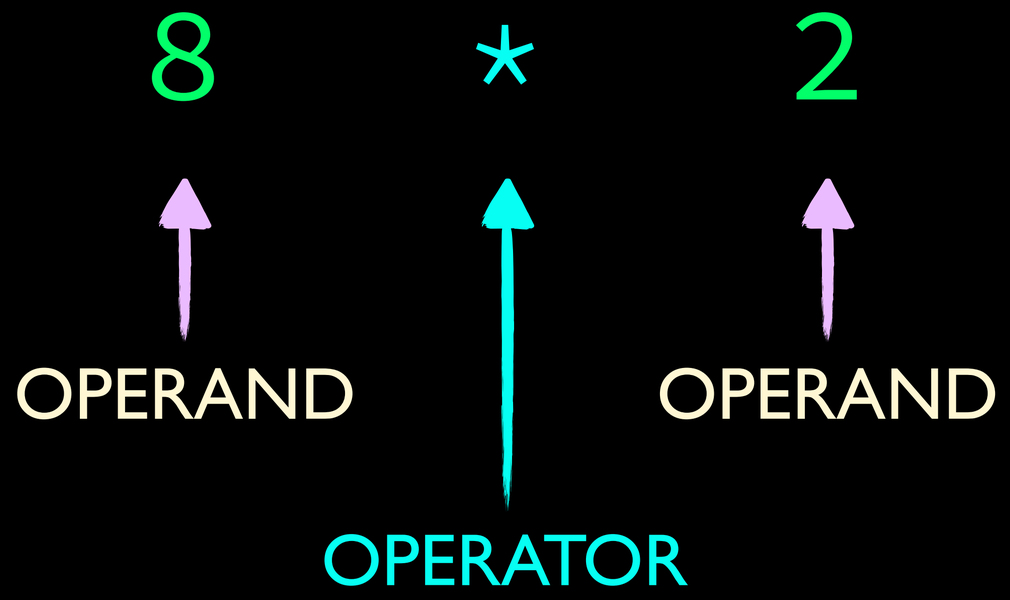
NUMBERS

Time to learn about basic numerical operations!



ARITHMETIC OPERATORS

1..2..3..go!

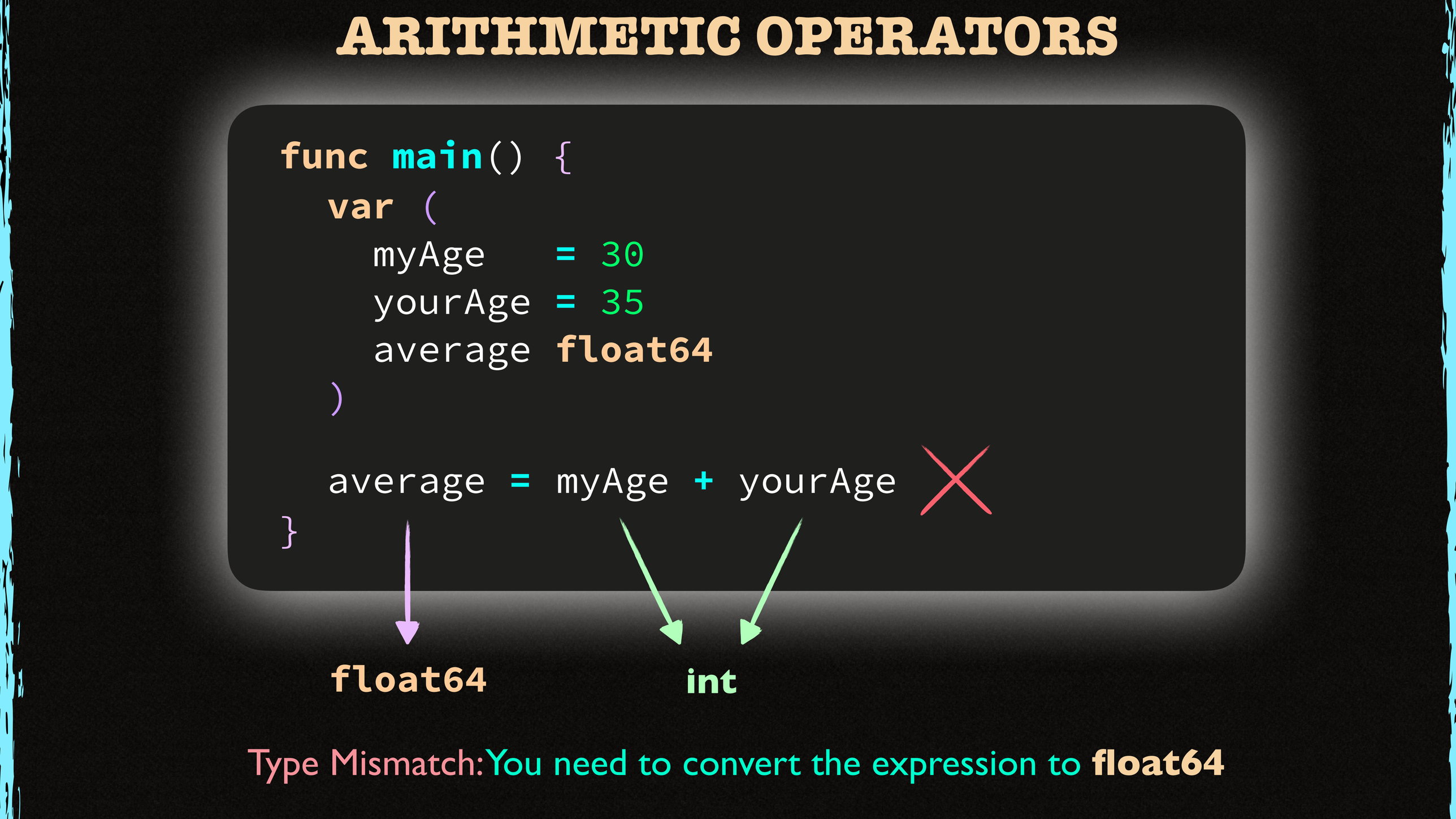

8
*
2
OPERAND
OPERAND
OPERATOR

ARITHMETIC OPERATORS

name	operator	expression	result
negation	`-`	`-(-2)`	`2`
product	`*`	`8 * -4.0`	`-32.0`
quotient	`/`	`-4 / 2`	`-2`
remainder *(only works w/integers)*	`%`	`5 % 2`	`1`
sum	`+`	`1 + 2.5`	`3.5`
difference	`-`	`2 - 3`	`-1`

ARITHMETIC OPERATORS

```
func main() {
    var (
        myAge    = 30
        yourAge = 35
        average float64
    )

    average = myAge + yourAge
}
```



Type Mismatch: You need to convert the expression to **float64**

ARITHMETIC OPERATORS

```go
func main() {
    var (
        myAge   = 30
        yourAge = 35
        average float64
    )

    average = float64(myAge + yourAge) / 2

    fmt.Println(average)
}
```

```
$ go run main.go
32.5
```

FLOAT INACCURACY

```go
func main() {

    ratio := 1.0 / 10

    fmt.Printf("%.60f", ratio)

}
```

```
$ go run main.go
0.1000000000000000055511151231257827021181583404541015625000000
```

PRECEDENCE

determines the order of calculations

PRECEDENCE

name	expression	same as	result
multiplication	`2 + 2 * 4 / 2`	`2 + ((2 * 4) / 2)`	`10`
addition	`1 + 4 - 2`	`(1 + 4) - 2`	`3`
change it!	`(2 + 2) * 4 / 2`	`((2 + 2) * 4) / 2`	`8`

PRECEDENCE

example

```go
func main() {
  n, m := 1, 5

  fmt.Println(2 + 1 * m / n)
  fmt.Println(((2 + 1) * m) / n)
}
```

```
$ go run main.go
7
15
```

*1 * m = 5*

5 / n = 5

5 + 2 = 7

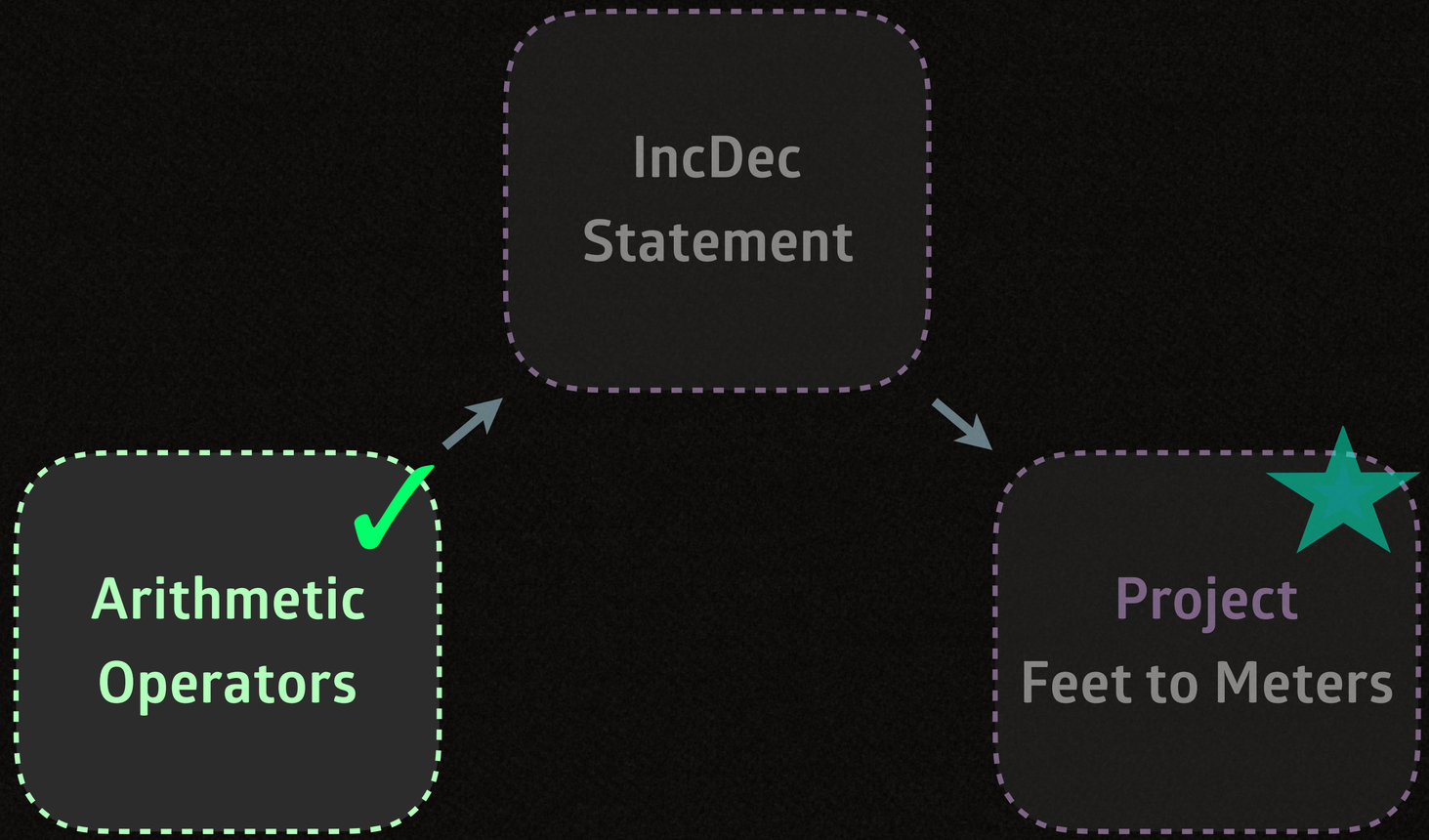
2 + 1 = 3

*3 * 5 = 15*

15 / 1 = 15

ARITHMETIC OPERATORS

Congrats!



INCDEC STATEMENT

expression++ expression--

INCDEC STATEMENT

Increases or decreases numeric values by one

```go
func main() {
  var n int

  n = n + 1
  fmt.Println(n)

  n += 1
  fmt.Println(n)

  n++
  fmt.Println(n)
}
```

incdec statement

`n++`	✓
`n--`	✓
`5 + n--`	✗

```
$ go run main.go
1
2
3
```

INCDEC STATEMENT

Increases or decreases numeric values by one

```go
func main() {
    n := 10

    n = n - 1
    fmt.Println(n)

    n -= 1
    fmt.Println(n)

    n--
    fmt.Println(n)
}
```

```
$ go run main.go
9
8
7
```

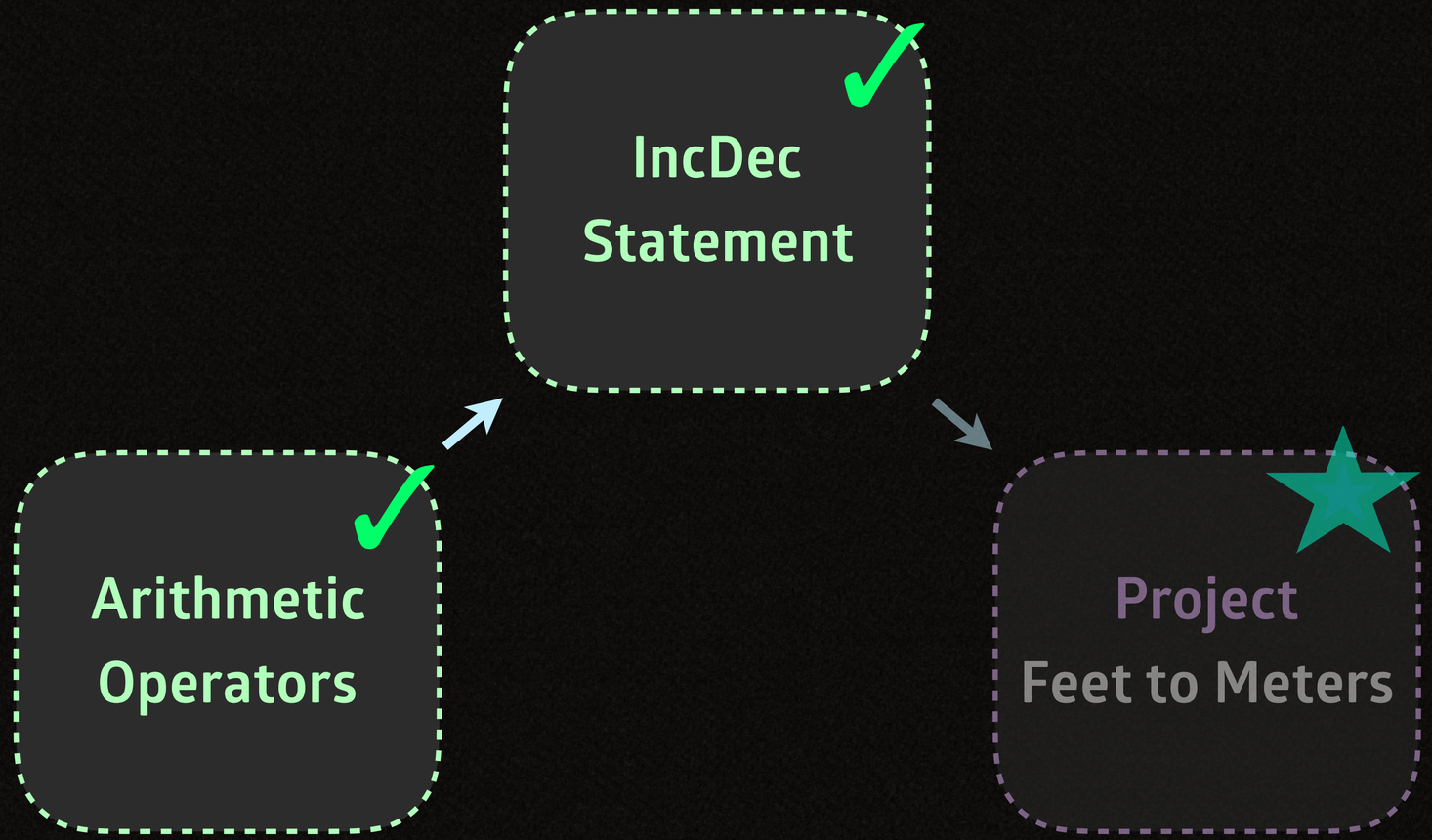
ASSIGNMENT OPERATIONS

variable *= 2

variable = variable * 2

ARITHMETIC OPERATORS

Done! Done!



FEET TO METERS

get and convert input from the command line

OS PACKAGE

strconv = string conversion

allows you to convert basic values from/to a string value

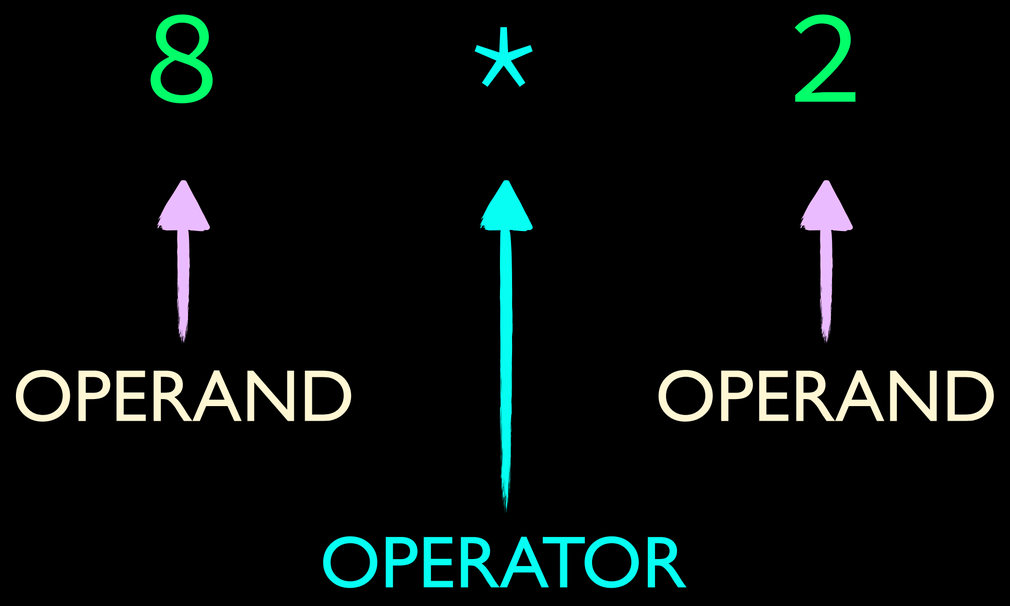
EXERCISE

convert celsius to fahrenheit

1. Get a number from the command line argument as Celsius
2. Convert it into a *float64* using *strconv*
3. Convert it to Fahrenheit
4. Print it

```
20°C × 1.8 + 32 = 68°F
```

summary


8
*
2
OPERAND
OPERAND
OPERATOR

arithmetic operators

```
-(-2)
```

unary operator

```
8 * -4.0
-4 / 2
5 % 2
```

multiplication operators

only works with integers

```
1 + 2.5
2 - 3
```

addition operators

floats are inaccurate

0.1000000000000005

int *and* float = float

2 + 2.0 = 4.0

2 + 2 = 4

assignment operations

variable += 10 → variable = variable + 10

variable *= 5 → variable = variable * 5

incdec statement

variable++ → variable += 1 → variable = variable + 1

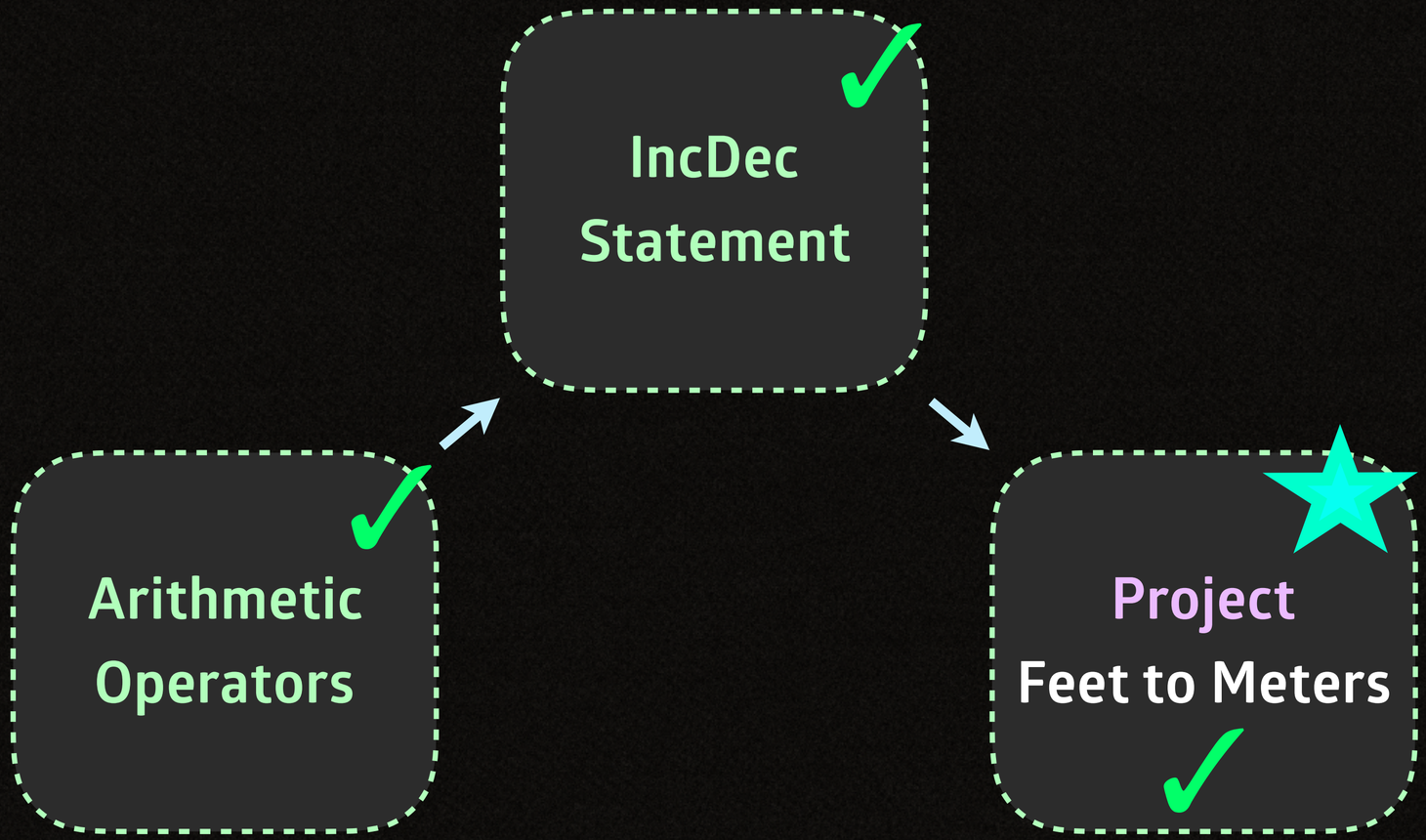
variable-- → variable -= 1 → variable = variable - 1

strconv package

ParseFloat("3.5", 64) = 3.5

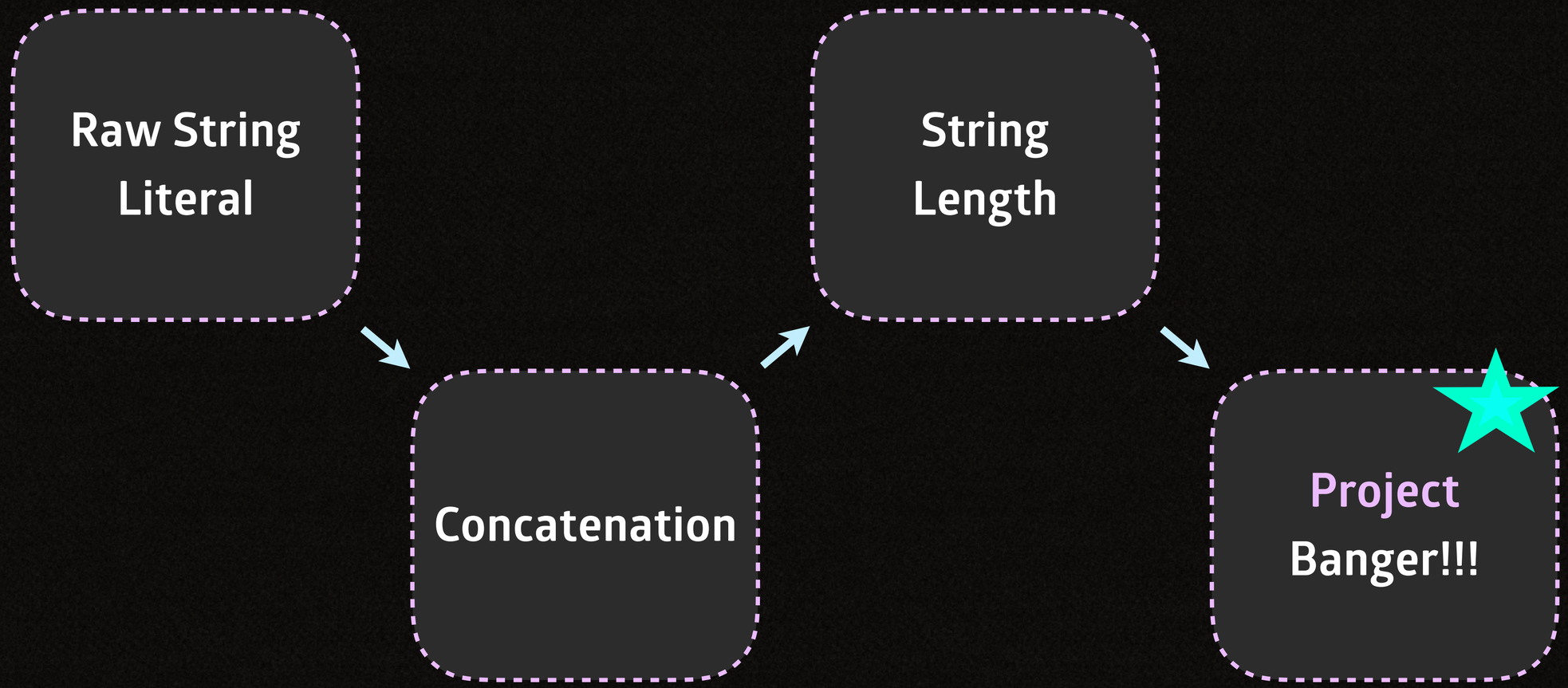
NUMBERS

Congrats!



BASIC STRINGS

Time to learn about the basics of strings!



RAW STRING LITERALS

let's investigate the raw string literal

TYPES

`string` → ***string literal***

"hi there 星!"

single-line

interpreted

`string` → ***raw string literal***

`hi
there!`

multi-line

not interpreted

Their type is ***string***

TYPES

`string` → ***string literal***

"hi there 星!"

single-line

processed

`string` → ***raw string literal***

`` `hi
there!` ``

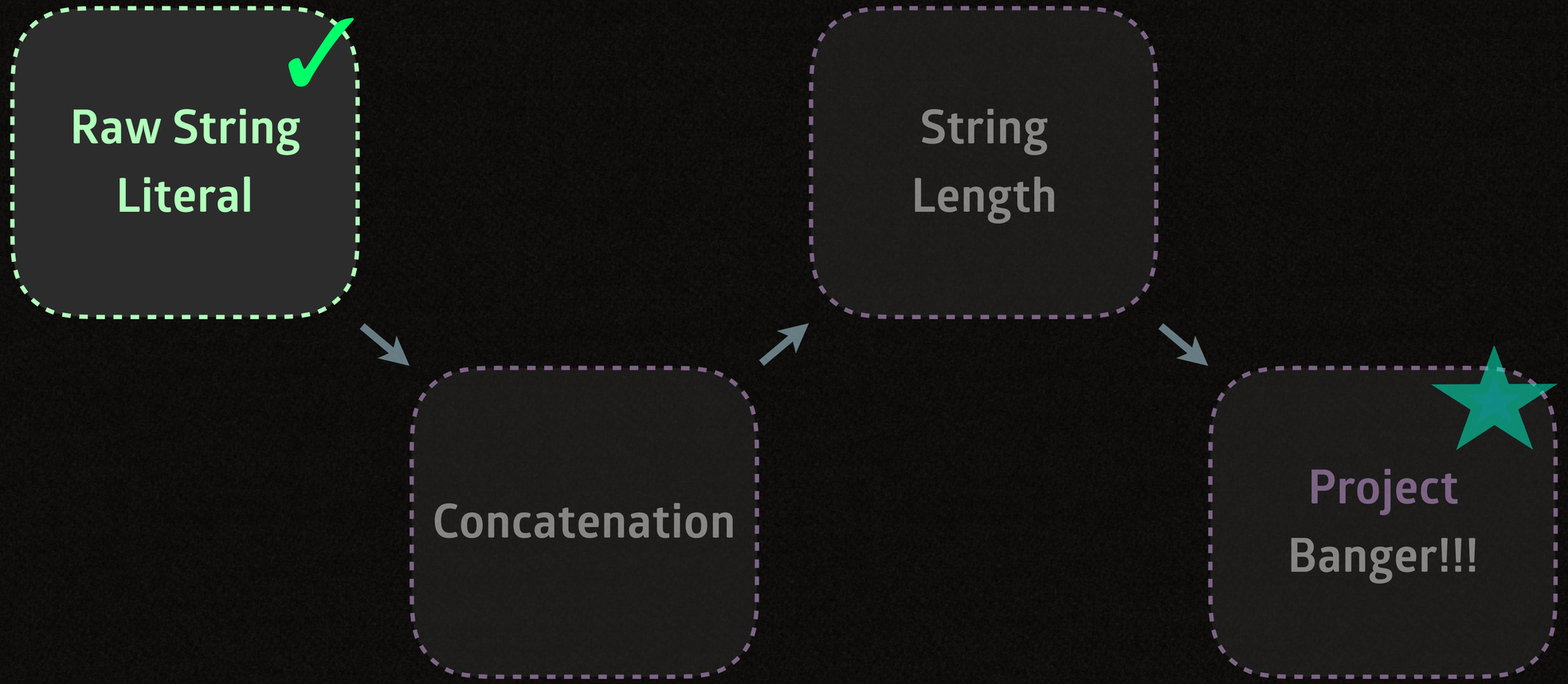
multi-line

unprocessed

Their type is ***string***

BASIC STRINGS

Congrats!



STRING CONCATENATION

combine multiple string values together

STRING CONCATENATION

Concatenation operator + combines multiple strings together and creates a new string value

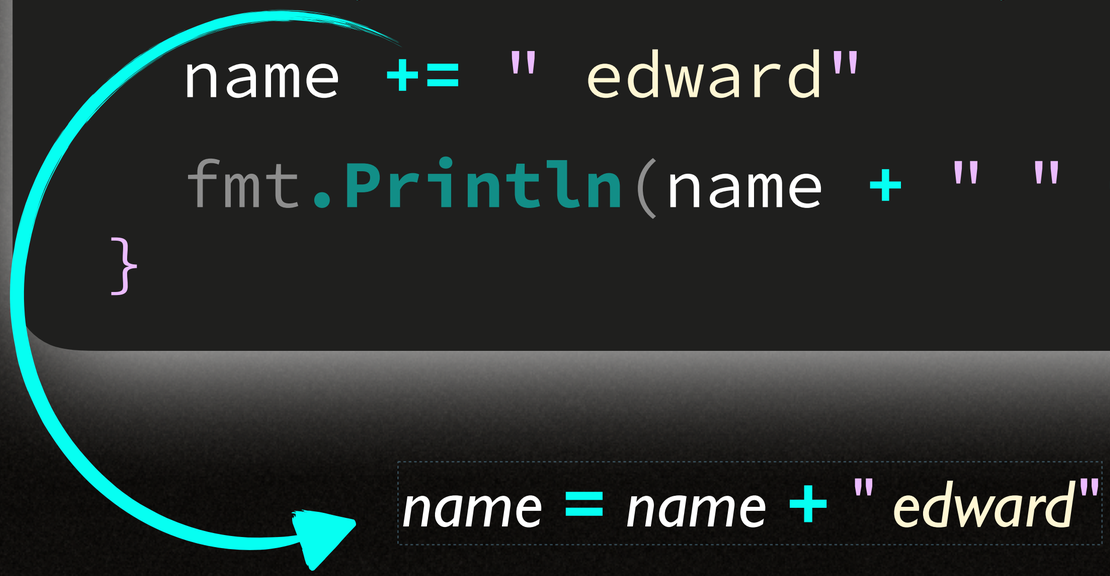
```go
func main() {
  name, last := "carl", "sagan"

  fmt.Println(name + " " + last)
}
```

```
$ go run main.go
carl sagan
```

STRING CONCATENATION

Concatenation operator can be used in an assignment operation

```go
func main() {
  name, last := "carl", "sagan"
  name += " edward"
  fmt.Println(name + " " + last)
}
```

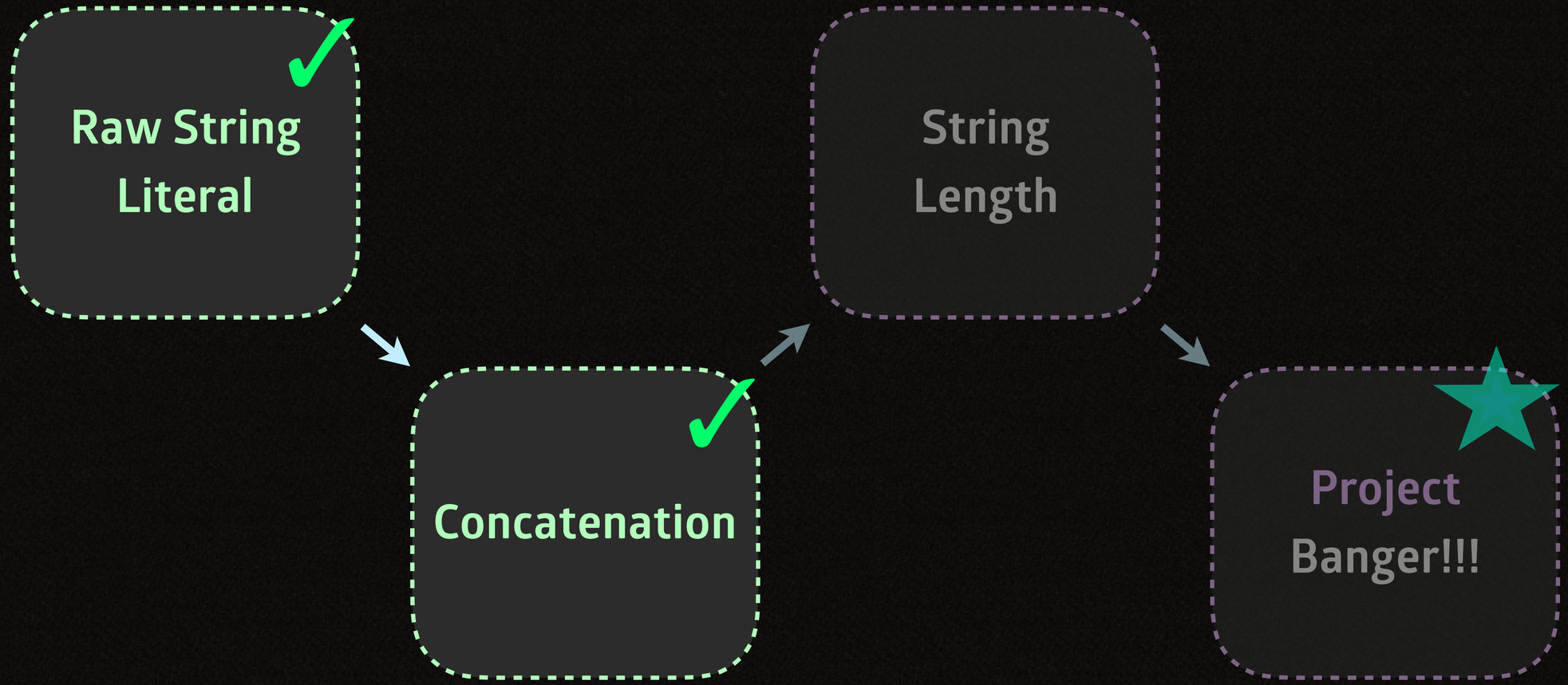


name = name + " edward"

```
$ go run main.go
carl edward sagan
```

BASIC STRINGS

Congrats!



STRING LENGTH

get the length of a string value

STRING LENGTH

len *built-in function returns the length of a string value in bytes*

```
func main() {
  name := "carl"

  fmt.Println(
    len(name))
}
```

`len(`*stringValue*`)`
length of a string

```
$ go run main.go
4
```

len *returns how many bytes in a string*

STRING LENGTH

Unicode characters can be 1 to 4 bytes each

```go
func main() {
    name := "İnanç"

    fmt.Println(
        len(name))
}
```

5 characters

'İ' = 2 bytes
'n' = 1 byte
'a' = 1 byte
'n' = 1 byte
'ç' = 2 bytes

`len(stringValue)`

length of a string in bytes

```
$ go run main.go
7
```

STRING LENGTH

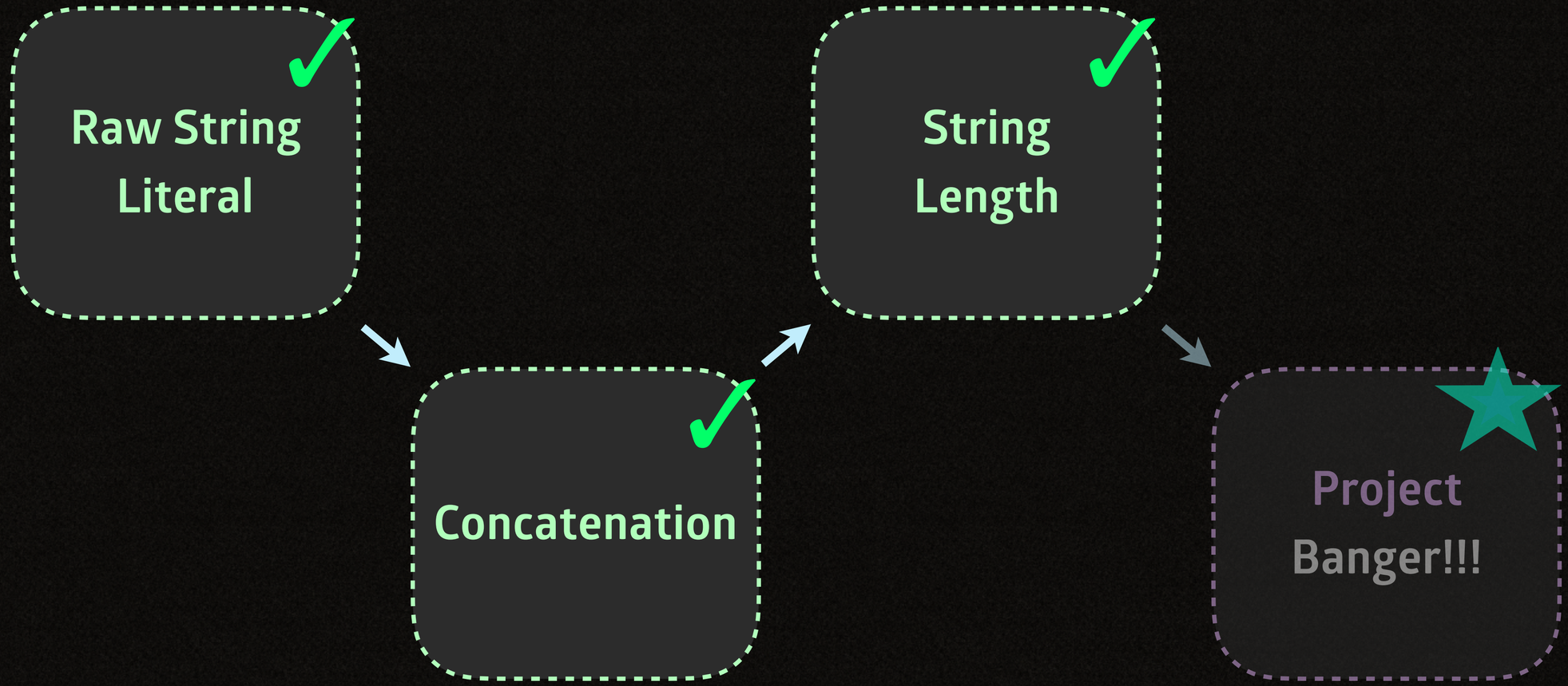
*utf8 package's **RuneCountInString** function finds the number of characters in a string*

```
import ( "fmt"; "unicode/utf8" )

func main() {
   name := "İnanç"

   fmt.Println(
      utf8.RuneCountInString(name))
}
```

```
$ go run main.go
5
```

BASIC STRINGS

Congrats!



BANGER

get input + bang it

go run main.go hey
HEY!!!

go run main.go hello
HELLO!!!!!

OS PACKAGE

strings package

provides utility functions for manipulating strings

https://golang.org/pkg/strings/

EXERCISE

Also add "!" in front of the passed argument

go run main.go hey
!!!HEY!!!

go run main.go cool
!!!!COOL!!!!

🎃 **BONUS:** Use the **Repeat** function **only once**

summary

raw string literals

```
`{
   "key": "value"
}`
```

concatenation operator

"Good" + " " + "Bye"

"Good Bye"

string conversion

strconv.Itoa(255) = "255"

length of strings

returns the **number of bytes** in a string value

len("İnanç") **=** 7

len("hey!") **=** 4

len("hi!") **=** 3

string literals are utf-8 encoded

```
utf8.RuneCountInString("İnanç") = 5

len("İnanç") = 7
```

BASIC STRINGS

Congrats!